"Express Mail" mailing label number:

EV 335894744 US

MULTI-LAYER ARCHITECTURE FOR PROPERTY MANAGEMENT

Letian Chen
John R. Quagliata
Jason R. Hull
Sean M. Le
David M. Bogue

BACKGROUND

Field of the Disclosure

[1001] The present disclosure relates to methods and systems for managing properties.

Description of the Related Art

[1002] Many Web applications use properties to allow application data to be accessible and easily updated. As used in this patent application, a property includes a name-value pair that resides externally to application code. Properties are used to initialize variables whose values are subject to change, and therefore should not be hard coded into a computer program.

[1003] Properties are well-suited to store values that may change over time such as Uniform Resource Locators (URLs) and error messages. For example, a URL may change because a service moves to a new machine, or an error message may change in response to a client request. Properties are also well-suited for values that may differ for each instance of an application. For example, a test application instance may use different URLs than a production application instance.

[1004] There are two standard strategies for using properties in Web applications. A first strategy is to use property files, examples of which include Java properties files and .INI files. A second strategy is to store properties in a database. Both of these strategies are subject to some limitations.

[1005] As property files and tables are usually hard to read and not user friendly, manually editing properties is tedious and prone to error. Also, manually editing a file or table does not impose data validation on the property information. Further, manually editing properties can corrupt the file or database.

[1006] Directly editing property files and tables may involve accessibility and/or authorization issues. For example, editing properties that are deployed on a remote server may be difficult because of file permissions or database security. Furthermore, editing database tables requires someone with database experience and generally is difficult to perform by non-technical personnel.

[1007] In a conventional property system, properties are loaded only once, which means that properties cannot be updated while the application is running. For a property value to be changed, the property file or database must be updated and the application must be restarted. Thus, such properties are not inherently dynamic once an application is running.

[1008] To create dynamic properties, polling can be used to detect property changes. However, polling can waste system resources. For example, a significant amount of processing time may be spent needlessly reloading properties or scanning for properties that have changed. Polling imposes a trade-off between resources spent on constantly checking for updates and the timeliness of obtaining the update.

BRIEF DESCRIPTION OF THE DRAWINGS

[1009] The present invention is pointed out with particularity in the appended claims. However, other features are described in the following detailed description in conjunction with the accompanying drawing in which:

[1010] FIG. 1 is a block diagram of an embodiment of a multi-layer property manager system; and

[1011] FIG. 2 is a screen shot of an embodiment of the property management Web interface.

[1012] The use of the same reference symbols in different drawings indicates similar or identical items.

Description of the Preferred embodiment(s)

[1013] Embodiments of the present invention provide an efficient and effective system and method for property management in applications such as Web applications. The system provides maintainable, dynamic properties to Web applications while mitigating the shortfalls of conventional property files and databases.

[1014] FIG. 1 is a block diagram of an embodiment of a multi-layer property manager system. The system comprises three layers: a persistence layer 10, a Web interface layer 12, and a property manager application layer 14. The persistence layer 10 provides a storage medium 16 to store properties in either a property file or a database table. The persistence layer 10 can store additional information for each property. Examples of the additional information include, but are not limited to, date and/or time stamps indicating when the property was created and/or modified, author names indicating personnel who created and/or modified the property, descriptions or other comments associated with the properties, and version numbers associated with the property.

[1015] The Web interface layer 12 provides a Web-based Graphical User Interface (GUI) for users to work with the properties stored by the persistence layer 10. The Web interface layer 12 displays the names and values of properties in an easy-to-read, organized and searchable format. The Web interface layer 12 accepts user input to add, modify, delete and otherwise update the properties and their associated information stored by the persistence layer 10. The user input can be received via the Web, or in general, any computer network such as the Internet, an intranet, an extranet, a local area network or a wide area network. The Web interface layer 12 may be password protected to allow only specific users to log on and view/update parameters.

[1016] The persistence layer 10 is responsive to the Web interface layer 12 to update or otherwise modify the properties and their associated information based on the user input.

Further, the Web interface layer 12 notifies the property manager layer 14 that the properties have been changed.

[1017] The Web interface layer 12 enables properties to be viewed and modified without cumbersome file permissions or database access restrictions. Users need not be concerned about having database expertise or knowledge of a particular property file format.

[1018] The Web interface layer 12 can organize properties into categories. With this feature, users can view properties by category, and can add new properties to a category. This feature may be especially useful in applications with a large number of properties.

[1019] The property manager application layer 14 offers services to one or more applications 20 that use the properties. The services are exposed to an application through an application programming interface (API) provided by the application layer 14. The services include retrieving properties from the persistence layer 10, subscribing to or otherwise receiving property updates from the Web interface layer 12, and notifying an application when a property update occurs.

[1020] When the property manager is initialized, some or all of the properties may be loaded into a memory 21 of the property manager application layer 14. In this way, the property manager application layer 14 does not always have to access the persistence layer 10 to retrieve properties. When the number of properties being managed is small, it is faster for all of the properties to be loaded into the memory 21 at initialization, and subsequently retrieved directly from the memory 21 when requested. When the number of properties being managed is significantly large, it may be beneficial to store only a subset of the properties into the memory 21 for subsequent retrieval.

[1021] The motivation for storing properties in the memory 21 is that it is much faster to retrieve information directly from the memory 21 than from the persistence layer 10. However, because the memory 21 has limited capacity, a large number of properties may not be capable of being stored in the memory 21 or may adversely affect system performance if stored in the memory 21. Therefore, storing only a subset of the

properties in the memory 21 provides fast access to the subset (which may comprise the most frequently used properties, for example) without degrading system performance.

[1022] At run time, an application queries the property manager application layer 14 for a specific property value. In one embodiment of the query, the application passes a property key associated with the property to the property manager application layer 14. When queried, the property manager application layer 14 determines if the key exists. If the key is found, a value corresponding thereto is retrieved from either the persistence layer 10 or the memory 21 and returned to the application. If the key is not found, an error message indicating that the key does not exist is sent back to the application.

[1023] Applications register with the property manager application layer 14 to be notified of property updates. In one embodiment, the application creates a property listener for this purpose. For example, a first application class 22 may have a first property listener 24, a second application class 26 may have a second property listener 30, and a third application class 32 may have a third property listener 34. In general, any number N of application classes may have an associated property listener.

[1024] Each property listener has a list of property keys of interest to its application. Each property listener is passed to the property manager application layer 14, which in turn registers the listeners. Once a listener has been registered, the property manager application layer 14 notifies the listener when one of the listener's properties has been modified using the Web interface layer 12.

[1025] In one embodiment, the property manager application layer 14 maintains a lookup table 36 that records, for each property key, a list of listeners that have registered to listen for that key. When a property is updated, the Web interface layer 12 communicates the property key to the property manager application layer 14. In response thereto, the property manager application layer 14 uses the lookup table 36 to notify all of the listeners that have registered for the property key of the update. In response to receiving notification of the update, the listener modifies the values of the program variables within the application that are dependent on the property.

[1026] The above services enable properties to be updated dynamically, i.e. the applications 20 need not be restarted to use updated property values. Further, the above services enable property updates to be made available to any of the applications 20 substantially immediately after a property is modified. The update is communicated to an application in an efficient manner that avoids costly techniques such as polling.

[1027] The Web interface layer 12 and the property manager application layer 14 hide the details of the persistence layer 10 from one or more applications 20 which use the property manager system. Beneficially, the applications 20 which use the property manager system need not know a file layout or table structure of the persistence layer 10.

[1028] The applications 20 do not directly access the persistence layer 10. The applications 20 access the properties from the persistence layer 10 via the property manager application layer 14 at run time.

[1029] Embodiments of the present invention can be used in a variety of software applications. In one particular embodiment, the property manager is used with a middleware application that provides flow through ordering applications to at least one back-end application such as back-end legacy systems. A sample middleware application is a J2EE application that runs on the WebSphere® product from International Business Machines (IBM). The property manager provides a better way to handle properties than simply using JAVA .property files.

[1030] Further in the particular embodiment, the Web interface layer 12 may be implemented using JAVA Server Pages (JSPs) 40 and one or more servlets 42. For each property displayed by the interface, a description and time stamp are also displayed.

[1031] The property manager application layer 14 may be contained within a common project 44 within the middleware application (as depicted in FIG. 1). Alternatively, the property manager application layer 14 may be separate from the project, e.g. the property manager code may be contained in a client JAVA Archive (JAR) file, a Dynamic Link Library (DLL) file, a Microsoft Corporation's Cabinet (CAB) file, or another suitable file type for packaging.

[1032] Classes of the middleware application that are to receive property updates create a property listener object and register the listener with the property manager application layer 14.

[1033] The persistence layer 10 is implemented using a database such as one available from Oracle Corporation. A row in a property table stores a property key, property value, description and time stamp.

[1034] Embodiments of the herein-disclosed architecture can be implemented using computer-readable program code stored by a computer-readable medium. The computer-readable program code causes a computer system having one or more processors to provide the various blocks depicted in FIG. 1 and to perform the acts associated therewith. Examples of the computer-readable medium include, but are not limited to, a magnetic medium such as a hard disk or a floppy disk, an optical medium such as an optical disk (e.g. a CD or a DVD), and an electronic medium such as an electronic memory (e.g. a removable memory card or an integrated memory in the computer system).

[1035] FIG. 2 is a screen shot in an embodiment of the property management Web interface 12. The Web interface displays each property and its associated information in a corresponding row. Displayed in each row is a property name, a property value, an optional description associated with the property, and a date and time stamp associated with the property. For all of the displayed properties, the property names are displayed in a column 50, the property values are displayed in a column 52, the descriptions are displayed in a column 54, and the date and time stamps are displayed in a column 56.

[1036] Each property is user-selectable by an associated check box (a representative one being indicated by reference numeral 60). A delete button 62, when selected by a user, initiates deletion of one or more user-selected properties. An edit button 64, when selected by the user, initiates editing of one or more user-selected properties. An insert button 66, when selected by the user, initiates a process to facilitate a new property to be inserted.

[1037] The user can change his/her password to access the Web interface by selecting a hyperlink 70.

[1038] It will be apparent to those skilled in the art that the disclosed embodiments may be modified in numerous ways and may assume many embodiments other than the preferred forms specifically set out and described herein. For example, embodiments of the present invention may be used for property management in non-Web applications. Further, the Web interface layer 12 may be replaced by another user interface layer, such as a GUI, having substantially the same features.

[1039] The above disclosed subject matter is to be considered illustrative, and not restrictive, and the appended claims are intended to cover all such modifications, enhancements, and other embodiments which fall within the true spirit and scope of the present invention. Thus, to the maximum extent allowed by law, the scope of the present invention is to be determined by the broadest permissible interpretation of the following claims and their equivalents, and shall not be restricted or limited by the foregoing detailed description.